Numerical Algorithms Group NAGWare F77 Tools Evaluation

Terance L. Lam
Computer Sciences Corporation
Numerical Aerodynamic Simulation Division
NASA Ames Research Center

Report RND-91-014 December 1991

Abstract

This document discusses the evaluation of the Numerical Algorithms Group's NAGWare F77 Tools (Beta Release 1, NASG401N) on a SGI 4D workstation. Features of NAGWare F77 tools are discussed and compared to the Information Techniques Inc. FORTRAN-lint[1] Fortran Source Code Analyzer. A cost analysis and recommendations are included. It is recommended that neither the NAGWare F77 Tools nor the IPT Fortan-Lint be purchased due to the limited interest at NAS.

1.0 Introduction

The NAGWare F77 Tools are direct descents of the Toolpack tools which originated from an idea suggested at the Jet Propulsion Laboratories in Pasadena, California, in the autumn of 1978 [1]. The idea was to put together a comprehensive collection of mathematical software development tools. This project was supported in the U.S.A. by the National Science Foundation and the Department of Energy, and in the U.K. by the Science and Engineering Research Council. The aims of the project were:

- To provide a suite of tools to assist in the production, testing, maintenance and porting of medium sized mathematical software projects written in standard conforming Fortran 77.

[1] FORTRAN-lint is a registered trademark of Information Processing Techniques, Inc.

- To investigate the development of extensible programming support environments built around integrated tool suites.

The present study performed an evaluation of the NAGWare F77 Tools, compared its features with the IPT Fortran Source Code Analyzer, and determined which tool suite would be most beneficial to the NAS community. Hopefully, these tools can be used to help NAS in the development of truly portable Fortran 77 programs across different computer platforms such as the Cray[2] supercomputers and the SGI[3] and SUN[4] workstations.

2.0 User Interface

The NAGWare F77 tools use the conventional UNIX[5] user interface. These tools employ a set of scripts that hide any complexity of the tools. Default options have been chosen for the scripts. Invoking a NAGWare F77 Tools script requires only a command line with the script name followed by a list of the input files. Any necessary intermediate files are generated automatically by the scripts as temporary files. Upon successful completion, each input file is backed up to file.orig and is replaced by the polished output.

3.0 Documentation

The NAGWare F77 Tools provide complete on-line manual pages and a 40-page manual [1]. The manual includes a tutorial section which guides novice users through the tools. Some examples in the tutorial were used in the evaluation process.

4.0 Installation

The NAGWare F77 tools came in as a tar tape. The contents of the software are organized as in figure 1.

[2] Cray is a registered trademark of Cray Research, Inc.
[3] SGI is a registered trademark of Silicon Graphics, Inc.
[4] SUN is a registered trademark of SUN Microsystems, Inc.
[5] UNIX is a registered trademark of AT&T.

	|---- bin	all executable
	|---- scripts	shell scripts calling the executable in bin
NAGWare_f77_tools	|---- examples	tutorial examples; should be accessible to users
	|---- results	results used for installation verification
	|---- test	installation test script
	|---- manl	manual pages

Figure 1. NAGWare F77 Tools

The installation involved loading the tape using the standard tar procedure and moving the executable to the appropriate directory (e.g. /usr/local/bin). The complete installation procedures were:

- cd /usr/local/bin; tar -xvf /dev/rst8
- for each shell script in NAGWare_f77_tools/scripts directory, modify the variable *toolexe* as

  toolexe="/usr/local/NAGWAre_f77/bin"
- add /usr/local/NAGWAre_f77/bin to the PATH environment
- add /usr/local/NAGWAre_f77/scripts to the PATH environment variable
- add /usr/local/NAGWAre_f77/manl to the MANPATH environment variable
- execute /usr/local/NAGWAre_f77/test/test_installation to verify the installation.

These installation procedures were relatively easy and succeeded at the first trial. An installation script could have made installation easier, because changing the same variable in all script files under NAGWare_f77_tools/scripts is more time consuming than changing one environment variable in an installation script.

5.0 Functional Descriptions

The NAGWare F77 Tools are divided into three classes; editing tools, transformation tools and analysis tools. Features of the NAGWare F77 Tools and their functionalities are discussed below. Details of these tools can be found in the manual pages. Simple examples are used to demonstrate the functionality of the NAGWare software in these sections. Some of these examples are quoted from the NAGWare F77 Tools' User's Guide.

5.1 Options File Editor

Nag_polopt is the NAGWare F77 Tools Polish Option File Editor. This menu-driven editor is used to create a polish options file which is used by nag_polish for controlling the appearance of all the transformation output. Figure 2 lists all control file options.

Basic Formatting Options	Customization Options	Conversions Options
Margin Control	Margin Control	Case Conversion
Re-labelling Format	Label Formatting	Include Files
CONTINUE Statement Insertion	Declaration Line-up	FORMAT Conversion Parameters
Move FORMAT Statements	Blank Character	
Sequence Number Format	Comment input	
Progress Trace	Comment Decoration	
Error Message	Declaration Line-Up	

Figure 2. Nag_polish Control Options

5.2 Transformation Tools

The transformation tools perform automatic conversions of Fortran 77 code. These tools are particularly useful in source code restructuring and standardization.

5.2.1 Precision Transformation

The Arithmetic Precision Transformation tool, *nag_apt*, converts single precision programs to double precision or vice versus. Two examples are shown below to demonstrate the transformations and their results. Figure 3 is the original Fortran 77 program used in this demonstration. This program contains implicitly and explicitly declared single precision variables, single precision constraints, a single precision specific intrinsic function, and the E format edit descriptor.

```
      PROGRAM EXAMPLE1

      REAL RADIUS, AREA, CIRCUM
      PI = 3.14159
      PRINT 9000
      READ *, RADIUS
      AREA = PI*RADIUS**2
      CIRCUM = 2.*PI*RADIUS
      PRINT 91000, CIRCUM, AREA, AMAX1(AREA,CIRCUM)
9000  FORMAT ('INPUT RADIUS:')
9100  FORMAT (CIRCUMFERENCE = ',F10.6,' AREA = ', E12.6,' MAX = ' E12.6)
      END
```

Figure 3. Example 1

Figure 4 is the output of a single-to-double precision transformation applied to EXAMPLE1 in figure 3. Constants and variables such as PI and RADIUS etc. are converted into double precision. AMAX1 has been changed to DMAX1 which is in double precision format. The last change was the FORMAT statement which changed the E descriptors to D descriptors.

```
      PROGRAM EXAMPLE1

      DOUBLE PRECISION PI
      DOUBLE PRECISION RADIUS, AREA, CIRCUM
      PI=3.14159D0
      PRINT 9000
      READ *, RADIUS
      AREA = PI*RADIUS**2
      CIRCUM = 2.D0*PI*RADIUS
      PRINT 91000, CIRCUM, AREA, AMAX1(AREA,CIRCUM)
9000  FORMAT ('INPUT RADIUS:')
9100  FORMAT (CIRCUMFERENCE = ',F10.6,' AREA = ', D12.6,' MAX = ', D12.6)
      END
```

Figure 4. Example 1 after Single-to-Double Precision Transformation.

Figure 5 is the double-to-single precision transformation on EXAMPLE1 in figure 4. All the double precision parameters are converted back to single precision. These two programs were compiled and verified that the functioning was the same. Of course, the output does not look the same as the original EXAMPLE1. The differences are the program organization and additional comments. This transformation tool can be applied to a group of software. The too automates the drudgery of an error-prone task.

```
      PROGRAM  EXAMPLE1

 C            ..  Local  Scalars  ..
              REAL  AREA,CIRCUM,PI,RADIUS
 C      ..
 C       ..  Intrinsic  Functions  ..
              INTRINSIC  AMAX1
 C      ..
              PI  =  3.14159
      PRINT  9000
      READ  *,RADIUS
              AREA  =  PI*RADIUS**2
              CIRCUM  =  2.*PI*RADIUS
              PRINT  9100,CIRCUM,AREA,AMAX1(AREA,CIRCUM)
 9000         FORMAT  ('  Input  Radius:')
 9100         FORMAT  ('  Circumference  =  ',F10.6,'   Area  =  ',D12.6,'   Max  =  ',
D12.6)
      END
```

Figure 5. Example 1 after Double to Single Precision Transformation.

5.2.2 Name Transformation

Nag_chname is the NAG F77 tools Fortran 77 name transformation tool. This tool systematically changes names in the Fortran 77 source file(s) according to commands from standard input or from an options control file. The control file (or standard input) is read for change requests of the form

<names> <comments><strings><hol><fold> <pat> = <rep>,

where <names>, <comments>, <strings>, <hol>, <fold> are either t (true) or f (false) indicating whether or not changes are to be applied to names, comments, strings, and holleriths respectively. <fold> is the flag for specifying whether case folding is required. <pat> is the replacement names or regular expression. Figure 6 is a log of the name change transformation session invoked with the pl_opt option. The Options File Editor started up and the control commands were entered. The original input file and the converted program are shown in figure 7 and 8 respectively. The transformation successfully changed SROTG and SAXPY to DROTG and DAXPY.

```
 %nag_chname  -po  pl.opt  example2.f
 Command:    ttttt         SROTG  =  DROTG
 Command:    ttttt  SAXPY  =  DAXPY
 Command:    EOF
```

Figure 6. Log of the Name Transformation Session.

```
PROGRAM EXAMPLE2

DOUBLE PRECISION X(2),Y(2)
DOUBLE PRECISION ALPHA,A,B,C,D
INTEGER N,INCX,INCY
EXTERNAL DROTG,DAXPY
CALL DROTG(A,B,C,D)
CALL DAXPY(N,ALPHA,X,INCX,Y,INCY)
END
```

Figure 7. Example 2

```
PROGRAM EXAMPLE2

DOUBLE PRECISION X(2),Y(2)
DOUBLE PRECISION ALPHA,A,B,C,D
INTEGER N,INCX,INCY
EXTERNAL SROTG,SAXPY
CALL SROTG(A,B,C,D)
CALL SAXPY(N,ALPHA,X,INCX,Y,INCY)
END
```

Figure 8. Example 2 after Nag_chname

5.2.3 Declaration Standardization

Nag_decs is the NAGWare Fortran 77 declaration standardization tool. This tool can rebuild the declaration section of a program. It may also be used to declare all implicitly typed variables. Figure 9 is the original source of EXAMPLE 3.

```
      PROGRAM EXAMPLE3

C     This is a simple program to show how nag_decs works.
      IMPLICIT DOUBLE PRECISION (D)
      IMPLICIT CHARACTER*3 (A-C, E)
      PARAMETER (F='TEST')
      INTEGER I,J
      COMMON /FRED/ A,B,C,D,F1,F2,F3 /FRED 2/ T
C     /FRED/ is a common block
      DIMENSION A(40)
      COMPLEX C(19*3+1, -9:6,9)
      CHARACTER*5 F1
      DATA ITWO /2/
C     Another comment
      I = ABS(K)
C     Comment in the code
      PRINT *, SQRT(I), F, J
      END
```

Figure 9. Example 3

In this example, nag_decs deleted then rebuilt the declarations as in figure 10. The output program shows that all variables have been explicitly declared. The comment that occurred in the declarations just after the common block declaration has been moved to the beginning of the declaration section; this is necessary because the declarations in the input program are deleted. All other comments retain their original position.

```
      PROGRAM EXAMPLE3

C     This is a simple program to show how nag_decs works.
C     /FRED/ is a common block
C       .. Parameters ..
      CHARACTER*(*) F
      PARAMETER (F='TEST')
C       ..
C       .. Scalars in Common ..
      DOUBLE PRECISION D
      REAL F2,F3,T
      CHARACTER*3 B
      CHARACTER*5 F1
C       ..
C       .. Arrays in Common ..
      COMPLEX C(19*3+1,-9:6,9)
      CHARACTER*3 A(40)
C       ..
C       .. Local Scalars ..
      INTEGER I,ITWO,J,K
C       ..
C       .. Intrinsic Functions ..
      INTRINSIC ABS,SQRT
C       ..
C       .. Common blocks ..
      COMMON /FRED/A,B,C,D,F1,F2,F3
      COMMON /FRED2/T
C       ..
C       .. Data statements ..
      DATA ITWO/2/
C       ..
C     Another comment
      I = ABS(K)
C     Comment in the code
      PRINT *,SQRT(I),F,J
      END
```

Figure 10. Example 3 after Declaration Transformation.

5.2.4 Fortran 77 Pretty Printer

Nag_polish , called the "polisher," is the NAGWare F77 Tools pretty printer. This tool reformats a Fortran 77 program according to a standard layout. A number of parameters can be set by users to control the appearance of the reformatted program. A polish control file may also be created using the nag_polopt command.

Users can configure almost every aspect of spacing within statements and between statements. Indentation, re-labelling and the breaking of long statements into continuation lines is handled automatically according to default or user supplied criteria. This tool can optionally add labelled CONTINUE statements to terminate every DO loop. The position and labelling of FORMAT statements can also be specified.

```
     PROGRA    M    EXAMPLE4
     D0 99 I   =1,  10
          J =I
99             CONTINUE
                          END
```

Figure 11. Example 4

```
     PROGRAM EXAMPLE4
     D0 99 I =1, 10
          J =I
99   CONTINUE
     END
```

Figure 12. Example 4 Output

The input in figure 11 is transformed to figure 12 by nag_polish. The original program was restructured into an easy to understand program using the default layout.

5.2.5 Fortran 77 Structuring Program

Nag_struct is the NAGWare F77 restructure tool. Its purpose is to rebuild the flow of control, within each processed program unit, to a standardized form based on the structuring algorithm described in [2].

The restructurer rebuilds the control flow of a program unit creating block IF constructs wherever possible. It may also optionally create DO WHILE constructs. This tool also makes Fortran 66 code look much like Fortran 77 and hence enhance the control flow of the program.

```
      PROGRAM EXAMPLE5

 1    IF (A) THEN
         PRINT *,B
            IF (C) GOTO 2
         PRINT *,C
            GOTO 1
      END IF
 2       PRINT *,D
      END
```

Figure 13. Example 5

The following examples show how nag_struct improves the control flow of EXAMPLE #5. Figure 14 is the output of invoking nag_struct with default structuring options. The output has one GOTO statement eliminated. This has been achieved by negating the test in the IF statement in line 3 of the input program, which is line 4 of the output program, and making a block IF.

```
      PROGRAM EXAMPLE5

 10   CONTINUE
      IF (A) THEN
          PRINT *,B
          IF (.NOT.C) THEN
              PRINT *,C
              GO TO 10
              END IF
          END IF
          PRINT *,D
          END
```

Figure 14. Example 5 Restructured with Default Options.

```
      PROGRAM EXAMPLE5

      DO WHILE (A)
          PRINT *,B
          IF (C) THEN
             GO TO 10
          ELSE
             PRINT *,C
             END IF
          END DO
 10   PRINT *,D
     END
```

Figure 15. Example 5 Restructured with -while Option

Figure 15 is nag_struct output invoked with the -while option. With this option, a more elegantly structured program is produced by introducing the DO WHILE construct. But this DO WHILE construct is not ANSI standard F77 and may not be acceptable to some compilers.

5.3 Analysis Tools

The principal analysis tools verify Fortran 77 code against the ANSI standard and highlight non-portable usage of Fortran 77 features. These analysis tools are extremely useful tools in the verification of portable Fortran 77 programs.

5.3.1 Fortran 77 Portability Verifier

The Portability Verifier, *nag_pfort*, checks Fortran 77 source code against the ANSI standard and reports on non-portable usage of Fortran 77 features. It checks for:

- conformance to the NAGWare F77 Tools language standard
- conformance to the ANSI Fortran 77 standard
- conformance to a portable subset of the ANSI Fortran 77 standard, PFORT-77
- correct inter-program-unit communication
- unsafe references.

The analysis is carried out in two stages. Firstly, conformance to the NAGWare F77 Tools Fortran standard is checked (see manual page nag_Fortran77 for details). If this preliminary check is satisfied, nag_pfort proceeds to portability analysis in which the checks include additional Fortran 77 rules, conformance to PFORT-77, inter-program-unit communications, and unsafe references (see manual page nag_pfort for details). The second stage issues a portability report based on the PFORT-77 subset.

The Eagle Grid Generator Program was used in this analysis. The source consists of a total of 24,087 line of Fortran 77 code. The portability analysis was invoked with default options. Nag_pfort reported 20 informational messages, 109 warnings and 4 errors. These messages are summarized in figure 16. These results concurred with the Flint analysis and were verified legitimate.

Message Type	# of Message	Message
Informational	17	Conditional expression is constant
	3	Logical IF expression is constant
Warning	94	Unreferenced label
	15	Unused common block
Error	2	COMMON may only be initialized in BLOCK DATA
	1	Unexpected end of file (in main program)
	1	Syntax error

Figure 16. The Eagle Grid Generation Program Portability Analysis Summary

5.3.2 Call Tree Generator

Nag_fcalls, the call tree generator, prints the call graph of a Fortran 77 program. It also optionally lists which routines call each routine. Figure 17 is the input to nag_fcalls while Figure 18 is the output of the call tree analysis.

```
PROGRAM TEST       SUBROUTINE A(B)     SUBROUTINE C()      SUBROUTINE X()
EXTERNAL A, X      EXTERNAL B, C       PRINT *, 'C CALLED' PRINT *, 'X CALLED'
CALL A(X)          CALL C              END                 END
END                CALL B
                   END
```

Figure 17 Example 6

```
FCALLS          - FORTRAN/77 Call Tree Printer
        1       TEST
        2           A
        3               C
        4               X
```

Figure 18 Example 6 Nag_fcalls Output

This call-graph generation tool is very useful in software porting. This call-graph provides a means for the programmer to understand the program structure before the program can be compiled and executed. The call tree is verified functional, but there is one deficiency in this tool. This call tree generator requires that all errors be corrected before the call-graph can be generated. The Flint call tree generating software operates independently of the syntax checker. It performs call tree generation at the best of its capability even when syntax errors exist. This is an advantage of FORTRAN-lint over the NAGWare Tools.

6.0 IPT FORTRAN-lint versus NAGWare F77 Licensing Schemes

6.1 IPT FORTRAN-lint Licensing Schemes

As discussed in the [3], the IPT FORTRAN-lint's licensing scheme is based on the number of CPUs per Auditor File which reside on a CPU or a file server. This auditor file restricts a list of CPUs to access its utilities and the total number of concurrent Flint usage. For instance, a 4-user and 200-CPU auditor file allows 4 concurrent users among the 200 CPUs with permission to access the utility. Any workstation whose name is not included in this auditor file will be denied from accessing it. Figure 19 is a price list for FORTRAN-lint.

Since NAS has approximately 150 SGI workstations, the "per CPU" licensing scheme will cost NAS $585,000. The file server license scheme is a more feasible solution. To accommodate services for 150 CPUs, a 200-CPU and 16-user auditor file is required. The SPS will be used as the IPT Flint hosts. This license costs $21,900.

Number of CPUs	1 user	4 users	8 users	16 users	32 users
1	$3,900	$7,900	$9,900	$14,900	$19,900
up to 5	$4,900				
up to 8		$8,900			
up to 12	$5,900				
up to 16		$9,900	$11,900		
up to 25	$6,900	$10,900	$12,900	$17,900	
up to 50	$8,900	$12,900	$14,900	$19,900	
up to 100	$9,900	$13,900	$15,900	$20,900	$25,900
up to 200	$11,900	$14,900	$16,900	$21,900	$26,900
up to 300	$14,900	$18,900	$20,900	$25,900	$30,900

Figure 19. FORTRAN-lint Price Table (per auditor File)

An alternative offered by IPT is the SGI-Cray package. This license option, consisting of six 4-user SGI file server licenses and a 16-user Cray YMP license, allows 4 concurrent users on each SGI file server and 16 users on the YMP to use this software. IPT agrees to offer this package for $31,150, at a 44.3% discount of the full price ($47,400).

6.1 The NAGWare F77 Tools Licensing Schemes

There are two licensing schemes available on this product from NAG. Each scheme consists of an one-time initial license fee and a yearly maintenance fee thereafter. The yearly maintenance fee is 18% of the initial license cost. This maintenance fee will continue the technical support from NAG and automatic software upgrades. Figure 20 is the cost schedule for these schemes. Source license to this product is not an option.

The first one is the "per CPU" licensing scheme, in which the license fee depends strictly on the number of CPUs involved. One license costing $3,895 is required for each CPU. For example, if NAS has 115 SGI workstations, 115 licenses are required. The total license cost for 115 SGI workstations (with a volume discount) is $71,395.00 and the maintenance cost is $10,710.00 per year. Since there are more than 150 workstations at NAS, it is more cost effective to use the network server license scheme.

The cost of the network server license scheme is based on the number of network servers plus the number of concurrent users. A 20-concurrent-user license for one SGI workstation costs $10,877.00 ($3,895 + $6,982). The yearly maintenance cost is $1,957.86. This software is also available for the Cray YMP super computer, but in a different classification. The base price is offered at $6,995 and a 20-concurrent-user license costs $13,977.

License Type	Number of CPUs/Servers	1 user	< 5 users ($1,995)	< 10 users ($3,657)	< 15 users ($5,320)	< 20 users ($6,982)
SGI	1	$3,895	$5,890	$7,552	$9,215	$10,877
Network	2	$7,790	$9,785	$11,447	$13,110	$14,772
Server	3	$11,685	$13,680	$15,342	$17,005	$18,667
License	4	$15,580	$17,575	$19,237	$20,900	$22,562
Cray YMP	1	$6,995	$8,990	$10,652	$12,315	$13,977

Figure 20. NAGWare F77 Tools Price Schedule.

7.0 Recommendations

During the FORTRAN-lint evaluation, the SGI-Cray license package (six 4-user and unlimited-CPU SGI file server licenses and one 16-user Cray YMP license) offered by IPT costing $31,150 was recommended. This recommendation was made before the Support Processors Systems (SPS) and the Ultra Network were available. Users now can take advantages of the exceptional processing power of the SPS and the Ultra network on the SPS. Data can be transferred to the SPS over the UltrNet at a rate of 2 Gigabits per second for analysis on the SPS. The network transfer rate will no longer be an issue.

Although the FORTRAN-lint SGI-Cray package (as in the cost comparison, below) offers the lowest per-active-user cost, this scheme is not recommended. Using the Cray YMP as a processing unit for source code analysis is a waste of the computing resources of the supercomputers. The Cray computers should be used for more complex analysis. Fortran source code analysis should be performed on the SPS. Therefore, it is recommended that only the SPS should be considered as a candidate for these Fortran source code analyzers.

Software Package	Licensing Scheme	Total Cost	Per Active User Cost
FORTRAN-lint	One 16-user, 200-CPU SGI Server License	$21,900.00	$1,368.75
	One SGI-Cray licenses package at a 50% discount rate (six 4-user and unlimited-CPU SGI file server licenses and one 16-user Cray YMP license).	$31,150.00	$ 778.75
NAGWare	Four 15-user SGI Server License	$20,900.00	$1,306.25
	One SGI-Cray license package (4 15-user SGI server licenses and a 15-user Cray YMP license)	$33,115.00	$1,103.83

Figure 21. IPT Cost Schedule versus the NAGWare F77 Tools Cost schedule

Figure 21 shows that the total cost of the server license and the package licenses of FORTRAN-lint versus NAGWare are reasonably close; therefore, pricing is not the deciding factor. The functionality of each tool suite becomes more important.

A summary of the features and the costs of the NAGWare F77 Tools and the IPT FORTRAN-lint tools can be found in Figure 22. Please also refer to [3] for details of this product. The major differences between these tools are:

- The NAGWare tools differ from IPT Flint in that they provide transformation tools to convert Fortran 77 programs. These tools are extremely practical in porting, standardizing and restructuring of Fortran 77 programs. Although users can perform the same task using "awk," "sed" and the "vi" editor etc., these manual operations are prone to errors. The NAGWare transformation tools can be used to make repetitive changes to code easy, and eliminate errors introduced by "hand editing." This is the problem area of Flint. Flint only reports problems and does not provide tools to correct them. It relies on users to correct problems manually. This is the main disadvantage of FORTRAN-lint.
- Both the NAGWare tools and IPT FORTRAN-lint provide complete documentation, tutorials, and installation procedures. Both tools are easy to install and administrate.

	NAGWare F77 Tools	IPT FORTRAN-lint
Documentation	On-line manual pages and easy to follow tutorial	On-line manual pages and easy to follow tutorial
Installation	Easy, no installation tools required	Easy, equipped with installation script
Transformation Tools	• Pretty Printer • Restructurer • Precision Transformer • Name Changer • Declaration Standardizer • Options File Editor	None
Analysis Tools	Call Tree Analysis Portability Analysis (F77 ANSI X3.9-1978) • Syntax Analysis • Call interface Analysis • Data Usage Analysis	Call Tree Analysis Portability Analysis (F77 ANSI X3.9-1978) • Syntax Analysis • Call Interface Analysis • Data Usage Analysis • Symbols Table Analysis • Reference Table Analysis
Error messages	Appropriate	More Precise
Platforms	SGI, SUN & Cray YMP	SGI, SUN & Cray YMP
Licensing Scheme	Per CPU, per user	Per CPU, per user
Cost	$20,900	$21,900

Figure 22. Comparison of NAGWare F77 Tools and the IPT FORTRAN-lint

- On the other hand, IPT Flint produces detailed symbol tables and reference table for data reference checks; that is beyond the NAGWare tools' capability. These tables provide a comprehensive data usage and reference analysis, which helps users to locate data usage problems easily. The NAGWare tools do not provide these tools but still perform appropriate source code analysis.

- The NAGWare tools produce appropriate output messages, but the FORTRAN-lint messages are more accurate and direct. This is only a minor factor.

The NAGWare F77 Tools and the IPT FORTRAN-lint basically perform similar functions in Fortran 77 source code analysis. This analysis includes checks for syntax errors, data usage errors and interface errors. Both tools performed satisfactorily in the Fortran 77 portability check. The NAGWare F77 Tools out-performed the IPT FORTRAN-lint because of the transformation tools. Although the Flint symbols tables are particularly useful, experienced Fortran programmers could use the Fortran debugger for data usage checks effectively. Therefore, these symbols table and reference table are useful but not critical to users.

One difficulty in this evaluation is that NAS users' requirements on the Fortran development environment were not clear. In order to identify these requirements, several NAS scientists and Fortran programmers have been selected to participate in the NAGWare evaluation. Messages have also been broadcast on the SPS to welcome all other users' participation.

After a two-month evaluation period, it was found that the NAS users' general interest in these tools is limited. Only a few users have actually used it. The selected group of users have also reflected the same opinion. Users' comments on the NAGWare are:

> "The NAGWare Source Code Analyzer does not provide more help in development and debugging of Fortran software than the conventional Fortran compiler and debugger do."

> "The ideals of these NAGWare tools are promising but the software is unstable in its current state. It is not consistent in complex analysis. The NAGWare can improve in data checking, inter-program communication checking, floating point conversion and documentation. The capability of mixed languages analysis is helpful too."

> "Users may prefer a CASE solution, such as the Saber C which is a C development environment integrated with editor, debugger and Source

Code Control program. The CASE tool for the Fortran development environment should consist of an easy-to-use editor and a easy-to-use debugger, a Fortran Source Code Analyzer, as well as Configuration Management tools etc."

Because of the limited interest in the NAGWare F77 Tools, it is recommended that this software not be purchased. The reasons for the lack of interest in this product are that these tools do not meet the requirements of the users. Although there is a development interest expressed by the HSP group, it does not justify the acquisition of this product.

8.0 Conclusion

The evaluations of the NAGWare F77 Tools and the Information Processing Techniques FORTRAN-lint source code analyzer are finalized. These software packages perform similar tasks in the ANSI Fortran 77 source code conformance analysis. The IPT Flint is capable of producing detailed symbols table for data reference checks; while the NAGWare tools provide transformation tools which are particularly useful in source code restructuring and standardization.

Although these Fortran development tools seemed promising, an internal survey has shown that users' interest in these tools is limited. Therefore, it is recommended that neither the NAGWare F77 Tools nor the IPT Fortan-Lint be purchased unless a need for these tools is identified at a later time.

Reference

[1] "NAGWare F77 Tools", Numerical Algorithms Group, Inc., Commercial Publication, April 1991.

[2] Baker, Brenda S., "An Algorithm for Structuring Flowgraphs," [1977]. J. ACM 24, 1 (January 1977), 98-120.

[3] Lam, Terance "Information Processing Techniques FORTRAN-lint (Fortran 77 Source Code Analyzer) Evaluation", NASA Ames Report RND-91-013, June, 1991.